Understanding How They Attack Your Weaknesses: CAPEC

Robert A. Martin
Sean Barnum

May 2011
**Report Documentation Page**

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

<table>
<thead>
<tr>
<th>1. REPORT DATE</th>
<th>2. REPORT TYPE</th>
<th>3. DATES COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAY 2011</td>
<td></td>
<td>00-00-2011 to 00-00-2011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE AND SUBTITLE</th>
<th>5a. CONTRACT NUMBER</th>
<th>5b. GRANT NUMBER</th>
<th>5c. PROGRAM ELEMENT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding How They Attack Your Weaknesses: CAPEC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6. AUTHOR(S)</th>
<th>5d. PROJECT NUMBER</th>
<th>5e. TASK NUMBER</th>
<th>5f. WORK UNIT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</th>
<th>8. PERFORMING ORGANIZATION REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MITRE Corporation, 202 Burlington Road, Bedford, MA, 01730-1420</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</th>
<th>10. SPONSOR/MONITOR’S ACRONYM(S)</th>
<th>11. SPONSOR/MONITOR’S REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12. DISTRIBUTION/AVAILABILITY STATEMENT</th>
<th>13. SUPPLEMENTARY NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release; distribution unlimited</td>
<td>Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>14. ABSTRACT</th>
<th>15. SUBJECT TERMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16. SECURITY CLASSIFICATION OF:</th>
<th>17. LIMITATION OF ABSTRACT</th>
<th>18. NUMBER OF PAGES</th>
<th>19a. NAME OF RESPONSIBLE PERSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. REPORT</td>
<td>Same as Report (SAR)</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>unclassified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. ABSTRACT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unclassified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. THIS PAGE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unclassified</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std Z39-18
# Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Session Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00-8:45am</td>
<td>Software Security Knowledge about Applications Weaknesses</td>
</tr>
<tr>
<td>9:00-9:45am</td>
<td>Software Security Knowledge about Attack Patterns Against Applications</td>
</tr>
<tr>
<td>10:15-11:00am</td>
<td>Training in Software Security</td>
</tr>
<tr>
<td>10:15-11:00am</td>
<td>Software Security Practice</td>
</tr>
<tr>
<td>11:15-12:00am</td>
<td>Supporting Capabilities</td>
</tr>
<tr>
<td>11:15-12:00am</td>
<td>Assurance Cases</td>
</tr>
<tr>
<td>11:15-12:00am</td>
<td>Secure Development &amp; Secure Operations</td>
</tr>
</tbody>
</table>
The Long-established Principal of “Know Your Enemy”

“One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement.”

Chapter 3: “Planning the Attack”
- The Art of War, Sun Tzu
The Importance of Knowing Your Enemy

- An appropriate defense can only be established if you know how it will be attacked

- Remember!
  - Software Assurance must assume motivated attackers and not simply passive quality issues
  - Attackers are very creative and have powerful tools at their disposal
  - Exploring the attacker’s perspective helps to identify and qualify the risk profile of the software
What are Attack Patterns?

- Blueprint for creating a specific type of attack
- Abstracted common attack approaches from the set of known exploits
- Capture the attacker’s perspective to aid software developers, acquirers and operators in improving the assurance profile of their software
Leveraging Attack Patterns Throughout the Software Lifecycle

- Guide definition of appropriate policies
- Guide creation of appropriate security requirements (positive and negative)
- Provide context for architectural risk analysis
- Guide risk-driven secure code review
- Provide context for appropriate security testing
- Provide a bridge between secure development and secure operations
Common Attack Pattern Enumeration and Classification (CAPEC)

- Community effort targeted at:
  - Standardizing the capture and description of attack patterns
  - Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community
  - Gives you an attacker’s perspective you may not have on your own

- Excellent resource for many key activities
  - Abuse Case development
  - Architecture attack resistance analysis
  - Risk-based security/Red team penetration testing
  - Whitebox and Blackbox testing correlation
  - Operational observation and correlation

- Where is CAPEC today?
  - [http://capec.mitre.org](http://capec.mitre.org)
  - Currently 386 patterns, stubs, named attacks
Building software with an adequate level of security assurance for its mission becomes more and more challenging every day as the size, complexity, and tempo of software creation increases and the number and the skill level of attackers continues to grow. These factors each exacerbate the issue that, to build secure software, builders must ensure that they have protected every relevant potential vulnerability; yet, to attack software, attackers often have to find and exploit only a single exposed vulnerability. To identify and mitigate relevant vulnerabilities in software, the development community needs more than just good software engineering and analytical practices, a solid grasp of software security features, and a powerful set of tools. All of these things are necessary but not sufficient. To be effective, the community needs to think outside of the box and to have a firm grasp of the attacker’s perspective and the approaches used to exploit software.

Attack patterns are a powerful mechanism to capture and communicate the attacker’s perspective. They are descriptions of common methods for exploiting software. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.

To assist in enhancing security throughout the software development lifecycle, and to support the needs of developers, testers and educators, the Common Attack Pattern Enumeration and Classification (CAPEC) is sponsored by the Department of Homeland Security as part of the Software Assurance strategic initiative of the National Cyber Security Division. The objective of this effort is to provide a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. This site now contains the initial set of content and will continue to evolve with public participation and contributions to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community.

Release 1.6 Available
What do Attack Patterns Look Like?

- **Primary Schema Elements**
  - Identifying Information
    - Attack Pattern ID
    - Attack Pattern Name
  - Describing Information
    - Description
    - Related Weaknesses
    - Related Vulnerabilities
    - Method of Attack
    - Examples-Instances
    - References
  - Prescribing Information
    - Solutions and Mitigations
  - Scoping and Delimiting Information
    - Typical Severity
    - Typical Likelihood of Exploit
    - Attack Prerequisites
    - Attacker Skill or Knowledge Required
    - Resources Required
    - Attack Motivation-Consequences
    - Context Description

- **Supporting Schema Elements**
  - Describing Information
    - Injection Vector
    - Payload
    - Activation Zone
    - Payload Activation Impact
  - Diagnosing Information
    - Probing Techniques
    - Indicators-Warnings of Attack
    - Obfuscation Techniques
  - Enhancing Information
    - Related Attack Patterns
    - Relevant Security Requirements
    - Relevant Design Patterns
    - Relevant Security Patterns
Attack Pattern Description Schema

Formalization

Description

■ Summary

■ Attack_Execution_Flow
  - Attack_Phase^{1..3} (Name(Explore, Experiment, Exploit))
    ■ Attack_Step^{1..*}
      - Attack_Step_Title
      - Attack_Step_Description
      - Attack_Step_Technique^{0..*}
        ■ Attack_Step_Technique_Description
        ■ Leveraged_Attack_Patterns
        ■ Relevant_Attack_Surface_Elements
        ■ Observables^{0..*}
        ■ Environments
      - Indicator^{0..*} (ID, Type(Positive, Failure, Inconclusive))
        ■ Indicator_Description
        ■ Relevant_Attack_Surface_Elements
        ■ Environments
      - Outcome^{0..*} (ID, Type(Success, Failure, Inconclusive))
        ■ Outcome_Description
        ■ Relevant_Attack_Surface_Elements
        ■ Observables^{0..*}
        ■ Environments
      - Security_Control^{0..*} (ID, Type(Detective, Corrective, Preventative))
        ■ Security_Control_Description
        ■ Relevant_Attack_Surface_Elements
        ■ Observables^{0..*}
        ■ Environments

The HS SEDI FFRDC is managed and operated by The MITRE Corporation for DHS.
Blind SQL Injection

<table>
<thead>
<tr>
<th>Attack Pattern ID</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical Severity</td>
<td>High</td>
</tr>
</tbody>
</table>
| Description       | **Summary**

Blind SQL Injection results from an insufficient mitigation for SQL Injection. Although suppressing database error messages are considered best practice, the suppression alone is not sufficient to prevent SQL Injection. Blind SQL Injection is a form of SQL Injection that overcomes the lack of error messages. Without the error messages that facilitate SQL Injection, the attacker constructs input strings that probe the target through simple Boolean SQL expressions. The attacker can determine if the syntax and structure of the injection was successful based on whether the query was executed or not. Applied iteratively, the attacker determines how and where the target is vulnerable to SQL Injection.

In order to achieve this using Blind SQL Injection, an attacker:

For example, an attacker may try entering something like "username' AND 1=1; --" in an input field. If the result is the same as when the attacker entered "username" in the field, then the attacker knows that the application is vulnerable to SQL Injection. The attacker can then ask yes/no questions from the database server to extract information from it. For example, the attacker can extract table names from a database using the following types of queries:

"username' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 108".

If the above query executes properly, then the attacker knows that the first character in a table name in the database is a letter between m and z. If it doesn't, then the attacker knows that the character must be between a and l (assuming of course that table names only contain alphabetic characters). By performing a binary search on all character positions, the attacker can determine all table names in the database. Subsequently, the attacker may execute an actual attack and send something like:

"username'; DROP TABLE trades; --"
A Few Key Use Cases for CAPEC in Support of SwA

- Help developers understand weaknesses in their real-world context (how they will be attacked)
- Objectively identify specific attacks under which software must demonstrate resistance, tolerance and resilience for a given level of assurance
- Indirectly scope which weaknesses are relevant for a given threat environment
- Identify relevant mitigations that should be applied as part of policy, requirements, A&D, implementation, test, deployment and operations
- Identify and characterize patterns of attacks for security test case generation
- Identify and characterize threat TTPs for red teaming
- Identify relevant issues for automated tool selection
- Identify and characterize issues for automated tool results analysis
Where is CAPEC today?

• V1.4
  • Massive schema changes
    • Including addition of Observables structure
  • Some new content
  • Added initial set of network attack patterns

• V1.5
  • Added ~25 new network attack patterns
  • Added enhanced material to ~35 patterns
  • New View added for WASC Threat Taxonomy 2.0
  • Added ~65 mappings to CWE and several within CAPEC

• V1.6
  • Added 7 new application framework attack patterns as well as 68 new attack patterns in three new attack pattern categories: Physical Security Attacks, Social Engineering Attacks & Supply Chain Attacks
  • Added ~35 mappings to CWE and several within CAPEC

Currently 386 patterns, stubs, named attacks; 68 categories and 6 views
CAPEC Current Content
(15 Major Categories)

1000 - Mechanism of Attack
• Data Leakage Attacks - (118)
• Resource Depletion - (119)
• Injection (Injecting Control Plane content through the Data Plane) - (152)
• Spoofing - (156)
• Time and State Attacks - (172)
• Abuse of Functionality - (210)
• Exploitation of Authentication - (225)
• Probabilistic Techniques - (223)
• Exploitation of Privilege/Trust - (232)
• Data Structure Attacks - (255)
• Resource Manipulation - (262)
• Physical Security Attacks (436)
• Network Reconnaissance - (286)
• Social Engineering Attacks (403)
• Supply Chain Attacks (437)
CAPEC Current Content (Which Expand to...)

**1000 - Mechanism of Attack**

- Data Leakage Attacks - (118)
  - Data Excavation Attacks - (116)
  - Data Interception Attacks - (117)
- Resource Depletion - (119)
  - Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) - (82)
  - Resource Depletion through Flooding - (125)
  - Resource Depletion through Allocation - (130)
  - Resource Depletion through Leak - (131)
  - Denial of Service through Resource Depletion - (227)
- Injection (Injecting Control Plane content through the Data Plane) - (152)
  - Remote Code Inclusion - (253)
  - Analog In-band Switching Signals (aka Blue Boxing) - (5)
  - SQL Injection - (66)
  - Email Injection - (134)
  - Format String Injection - (135)
  - LDAP Injection - (136)
  - Parameter Injection - (137)
  - Reflection Injection - (138)
  - Code Inclusion - (175)
  - Resource Injection - (240)
  - Script Injection - (242)
  - Command Injection - (248)
  - Character Injection - (249)
  - XML Injection - (250)
  - DTD Injection in a SOAP Message - (254)
- Spoofing - (156)
  - Content Spoofing - (148)
  - Identity Spoofing (impersonation) - (151)
  - Action Spoofing - (173)
- Time and State Attacks - (172)
  - Forced Deadlock - (25)
  - Leveraging Race Conditions - (26)
  - Leveraging Time-of-Check and Time-of-Use (TOCTOU) Race Conditions - (29)
  - Manipulating User State - (74)
- Abuse of Functionality - (210)
  - Functionality Misuse - (212)
  - Abuse of Communication Channels - (216)
  - Forceful Browsing - (87)
  - Passing Local Filenames to Functions That Expect a URL - (48)
  - Probing an Application Through Targeting its Error Reporting - (54)
  - WSDL Scanning - (95)
  - API Abuse/Misuse - (113)
  - Try All Common Application Switches and Options - (133)
  - Cache Poisoning - (141)
  - Software Integrity Attacks - (184)
  - Directory Traversal - (213)
  - Analytic Attacks - (281)
  - Probabilistic Techniques - (223)

**Exploitation of Authentication - (225)**

- Exploitation of Session Variables, Resource IDs and other Trusted Credentials - (21)
  - Authentication Abuse - (114)
  - Authentication Bypass - (115)
- Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) - (232)
  - Privilege Escalation - (233)
  - Exploiting Trust in Client (aka Make the Client Invisible) - (22)
  - Hijacking a Privileged Thread of Execution - (30)
  - Subvert Code-signing Facilities - (68)
- Resource Depletion through Flooding - (125)
  - Resource Depletion through Allocation - (130)
  - Resource Depletion through Leak - (131)
  - Denial of Service through Resource Depletion - (227)
- Resource Depletion through Allocation - (130)
  - Resource Depletion through Leak - (131)
- Resource Injection - (240)
  - Script Injection - (242)
  - Command Injection - (248)
  - Character Injection - (249)
  - XML Injection - (250)
  - DTD Injection in a SOAP Message - (254)
- DTD Injection in a SOAP Message - (254)
  - Accessing/Intercepting/Modifying HTTP Cookies - (31)
  - Buffer Attacks - (123)
  - Attack through Shared Data - (124)
  - Integer Attacks - (128)
  - Pointer Attack - (129)
- Resource Manipulation - (262)
  - Accessing/Intercepting/Modifying HTTP Cookies - (31)
  - Input Data Manipulation - (153)
  - Resource Location Attacks - (154)
  - Infrastructure Manipulation - (161)
  - File Manipulation - (165)
  - Variable Manipulation - (171)
  - Configuration/Environment manipulation - (176)
  - Abuse of transaction data structure - (257)
  - Registry Manipulation - (269)
  - Schema Poisoning - (271)
  - Protocol Manipulation - (272)
- Network Reconnaissance - (286)
  - ICMP Echo Request Ping - (285)
  - TCP SYN Scan - (287)
  - ICMP Echo Request Ping - (288)
  - Infrastructure-based footprinting - (289)
  - Enumerate Mail Exchange (MX) Records - (290)
  - DNS Zone Transfers - (291)
  - Host Discovery - (292)
  - Traceroute Route Enumeration - (293)
  - ICMP Address Mask Request - (294)
  - ICMP Timestamp Request - (295)
  - ICMP Information Request - (296)
  - TCP ACK Ping - (297)
  - UDP Ping - (298)
  - TCP SYN Ping - (299)
  - Port Scanning - (300)
  - TCP Connect Scan - (301)
  - TCP FIN scan - (302)
  - TCP Xmas Scan - (303)
  - TCP NULL Scan - (304)
CAPEC Current Content (386 Attacks...)

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Description</th>
<th>Related CAPECs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phishing</td>
<td>Trick people into providing sensitive information</td>
<td>CAPEC-101, CAPEC-102</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>Access to unauthorized data</td>
<td>CAPEC-201, CAPEC-202</td>
</tr>
<tr>
<td>Cross-Site Scripting</td>
<td>Execute malicious code in user's browser</td>
<td>CAPEC-301, CAPEC-302</td>
</tr>
<tr>
<td>Remote Code Execution</td>
<td>Execute arbitrary code on the system</td>
<td>CAPEC-401, CAPEC-402</td>
</tr>
<tr>
<td>Buffer Overflows</td>
<td>Overwrite data on the system</td>
<td>CAPEC-501, CAPEC-502</td>
</tr>
</tbody>
</table>

The HS SEDI FFRDC is managed and operated by The MITRE Corporation for DHS.
Current Maturation Paths

- Extend coverage of CAPEC
- Improve quality of CAPEC
- Expand the scope of CAPEC
- Bridge secure development with secure operations
- Improve integration with other standards (MAEC, CEE, etc.)
- Expand use of CAPEC
CAPEC Future Plans

- V1.7 (within the next month or two)
  - Will flesh out ~30-40 stub patterns to full patterns
  - Will include existing content that has been refined for quality & consistency
  - Will incorporate initial use of the Observables sub-schema

- Strategic focus for the near to mid-term will be on utilizing CAPEC as a bridge between secure development and secure operations

- Continue expanding and refining content

- Continue expanding outreach and supporting CAPEC use

- Establish initial compatibility program
Questions?

sbarnum@mitre.org